# BIVPlaform

## Руководство разработчика по компоненте “Гибкая таблица”

2025

# Гибкие таблицы

## *Общие понятия*

## «Компонент» или «Компонента»?

Есть вот такой ответ на https://newslab.ru/article/198877 Очевидно, что слова «компонент» и «компонента» происходят от одного – латинского componens, в родительном падеже componentis – «составляющий», и означают составную часть чего-либо, целый элемент. Так чем же отличаются «компоненты» в мужском и женском роде? Если они подразумевают одно и то же, то не проще ли называть всё одним словом?

Хотя официально разница в употреблении нигде не зафиксирована, похоже на то, что в русском языке словом «компонент» обозначают нечто осязаемое, материальное (основной компонент смеси), тогда как «компонента» – понятие целиком абстрактное и используется, в основном, как математический термин (компонента вектора по оси абсцисс).

На основании такого объяснения предлагаю использовать слово «компонента» (женский род).

## *Назначение компоненты «Гибкая таблица»*

Компонента предназначена для отображения данных в табличных формах (журналах). Компонента имеет 2 взаимосвязанные части: бэкенд и фронтенд.

Цели создания компоненты:

- унификация табличных форм;
- снижение затрат на разработку табличных форм, достаточно подключить необходимые библиотеки/расширения на бэкенде, а также задействовать компоненту на фронтенде;
- единый пользовательский интерфейс с табличными данными;
- настройка представления таблицы под требования пользователя без программирования, а именно:
  - определение состава выводимых колонок из списка доступных;
  - определение ширины, формата данных, шрифта каждой колонки;
  - определение порядка сортировки данных;
  - определение полей, по которым будет производится ввод критериев для отбора данных в таблицу (динамические фильтры), сохранять значения этих критериев для последующих выборок;
  - переключение между несколькими настроенными представлениями.

## *Понятие настройки, её атрибутивный состав, структура*

Настройка гибкой таблицы определяет:

- состав выводимых для показа колонок;
- формат колонок: ширину, формат данных, шрифт, цвет;
- состав полей для отбора записей с допустимыми условиями (например, для строкового атрибута «серия договора страхования» можно определить следующие допустимые условия: «равно», «начинается с», «содержит»);
- состав полей, по которым производится сортировка выводимых данных;

- дополнительные критерии для отбора записей, которые может программно наложить пользовательский интерфейс (фронтенд);
- количество записей на странице, предельное количество выводимых записей.

Настройки и значения критериев, введённых пользователем, сохраняются в базе данных.

## *Типы пользователей*

При работе с настройками гибких таблиц существует два типа пользователей:

- Обычный пользователь. Использует функционал гибких таблиц (журналов) с настройками в процессе выполнения своих задач.
- Администратор настроек гибких таблиц. Задача администратора — подготавливать настройки пользователям, корректировать настройки. Администратор не работает в функционале других пользователей с их настройками. У него есть отдельный функционал по работе с настройками гибких таблиц и только. Администратор может:

  ◦ Добавлять/изменять/удалять настройки любых пользователей. ◦ Копировать настройки. ◦ Удалять значения фильтров для настроек.

Обычных пользователей можно объединять в группы для того, чтобы пользователи разделяли настройки (при этом они такие пользователи не могут редактировать сами).

## *Виды настроек*

Предусмотрены следующие виды настроек:

1. частная для пользователя. Настройка доступна только самому пользователю (и администратору для корректировки). Если пользователь создаёт настройку, то он создаёт именно частную настройку. Для редактирования пользователю доступны только свои частные настройки. В поле KIND в таблице CMN_GRID_VIEW хранится значение 1. В поле KIND_SPECIFIER в таблице CMN_GRID_VIEW хранится имя пользователя. Наименование настройки в рамках пользователя должно быть уникальным, т. е. нельзя создать 2 частные настройки для одного и того же пользователя с одинаковым наименованием.
2. общая для группы пользователей Настройка доступна всем пользователям группы для построения таблицы. Такая настройка создаётся/редактируется/удаляется только администратором. В поле KIND в таблице CMN_GRID_VIEW хранится значение 2. В поле KIND_SPECIFIER в таблице CMN_GRID_VIEW хранится имя группы пользователей. Наименование настройки в рамках группы пользователей должно быть уникальным.
3. общая для всех пользователей Настройка доступна вообще всем пользователям для построения таблицы. Такая настройка создаётся/редактируется/удаляется только администратором. В поле KIND в таблице CMN_GRID_VIEW хранится значение 3. В поле KIND_SPECIFIER в таблице CMN_GRID_VIEW хранится null. Наименование общей настройки должно быть уникальным.

Наименования настроек разных видов могут совпадать. Правила отбора (доступность) настроек в случае совпадающего наименования см. далее.

## *Доступность настроек обычному пользователю*

В табличном представлении пользователю доступны для выбора настройки:

- собственные частные;
- общие для групп пользователей, в которые входит пользователь;
- общие для всех.

Если наименования настроек разных совпадает, то отображается только одна настройка, которая имеет меньший вид (более приватная). Например:

- если есть общая и групповая настройки, то для пользователя будет доступна групповая;
- если есть групповая и частная настройки, то для пользователя будет доступна частная.

Таким образом обеспечивается кастомизация настроек под отдельные группы пользователей или одного пользователя.

Табличное представление открывается с настройкой по умолчанию (основной настройкой). Настройка по умолчанию должна быть одна для сущности и журнала и вида настройки. В случае наличия нескольких настроек по умолчанию разных видов приоритет отдаётся настройке, которая имеет меньший вид (более приватная).

## *Права на настройки*

Для того, чтобы отображать данные согласно настройке, не требуется каких-либо отдельных специфических прав. Доступность функционала с табличным представлением должна регулироваться другими правами, не относящимися к настройкам гибких таблиц.

Предусмотрены 2 права:

- Право на администрирование настроек. При наличии такого права пользователь является Администратор настроек гибких таблиц и имеет доступ к специальному интерфейсу по управлению настройками.
- Право на редактирование собственных частных настроек. В этом случае в табличных интерфейсах появляется возможность создания/редактирования/удаления собственных частных настроек. Такое право выдаётся «продвинутым» пользователям, которые занимаются какой-то аналитической деятельностью (например актуарий, андеррайтер).

## *Ограничения по количеству записей*

При отображении данных в гриде существует ограничение на количество выводимых записей. Данное ограничение реализовано в первую очередь с целью защиты СУБД от постоянных “тяжёлых” запросов, которые может создавать компонента Гибкая таблица. Оптимально, чтобы запросы к таблице с различными условиями, ограничивающими выборку, укладывались в индексы. Однако не всегда условия, которые выставил пользователь попадают в индексы таблицы (например, условие “содержит”). В этом случае запросы от журналов могут негативно сказаться на производительности СУБД, которая будет перегружена операциями чтения.

Кроме того, если в таблице хранится большое количество записей (миллионы и более), то нет смысла выводить все записи пользователю в таблицу. Пользователь не будет просматривать такое большое количество записей, и вероятно ему надо уточнить критерии отбора записей в таблицу.

Исходя из данных соображений в механизме гибких таблиц есть следующие особенности:

1. Не производится расчет общего количества записей, удовлетворяющих критериям.
2. При отображении данных в гриде в таблицу / экспорте данных выбирается

ограниченное количество записей (по умолчанию 500 / 25 000).
3. Значения по умолчанию можно переопределить в application.properties для: всех сущностей и журналов, или для конкретной сущности, или для конкретной сущности и конкретного журнала. Подробнее про параметры вида com.bivgroup.entityjournal.max.* см. ниже в **Параметры в application.properties**.

Если пользователю важно знать общее число записей (например, количество прикреплённых к ЛПУ в рамках договора), то:

1. Необходимо обеспечить, чтобы такой запрос имел оптимальный план, использовал индекс и не приводил к полному сканированию таблицы.

2. Такой запрос можно выполнить отдельно и вывести его результат рядом с гибкой таблицей.

# *Документация для Back-End разработчика*

## Подключение библиотеки в проект

### *Состав библиотек*

- entity-info - расширение для сбора описаний сущностей.
- entity-ext-info - дополнительная библиотека для сбора описаний о расширенных атрибутах. Описания хранятся в таблицах INS_HBDESCR, INS_HBPROPDESCR, INS_HBDATAVER. Значения расширенных атрибутов (как правило) хранятся в таблицах B2B_CONTREXT, INS_HBSTORE. Актуально только для проектов СБС на базе структуры таблиц B2B. В остальных случаях подключать библиотеку НЕ надо!
- entity-data-struct - структура (pojo-классы) самой настройки таблицы.
- entity-data - расширение по работе с данными.
- entity-journal-dao - объявления сущностей для хранения настроек гибких таблиц. Существует 2 версии: для PostgreSQL и Oracle.
- entity-journal-test-auth - тестовая реализация интерфейса авторизации для гибких таблиц в виде заглушки.
- entity-journal - надстройка над расширением по работе с данными, которая уже формирует данные в формате гибких таблиц.

### *Подключение библиотеки в проект*

В проекте сервиса необходимо подключить следующие зависимости:

```
<properties>
    <biv.lib.version>3.13-SNAPSHOT</biv.lib.version>
</properties>
<dependencies>
    <dependency>
        <groupId>com.bivgroup.lib</groupId>
        <artifactId>entity-info</artifactId>
        <version>${biv.lib.version}</version>
    </dependency>
    <dependency>
        <groupId>com.bivgroup.lib</groupId>
        <artifactId>entity-journal</artifactId>
        <version>${biv.lib.version}</version>
    </dependency>
</dependencies>
```

Для проектов СБС на базе структуры данных B2B необходимо подключить зависимость:

```
<dependencies>
    <dependency>
        <groupId>com.bivgroup.lib</groupId>
        <artifactId>entity-ext-info</artifactId>
        <version>${biv.lib.version}</version>
    </dependency>
</dependencies>
```

где biv.lib.version содержит актуальную версию библиотеки.

Наименование версии имеет следующий вид для релизных версий (пример):

- 3.13

SNAPSHOT версии имеют следующий вид (пример):

- 3.13-SNAPSHOT

В проекте, где объявляются сущности @Entity обязательно подключение jandex-maven-plugin.

```
<properties>
    <jandex-maven-plugin.version>1.2.2</jandex-maven-plugin.version>
</properties>
<build>
    <plugins>
        <plugin>
            <groupId>org.jboss.jandex</groupId>
            <artifactId>jandex-maven-plugin</artifactId>
            <version>${jandex-maven-plugin.version}</version>
            <executions>
                <execution>
                    <id>make-index</id>
                    <goals>
                        <goal>jandex</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
        <plugin>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>${compiler-plugin.version}</version>
        </plugin>
    </plugins>
</build>
```

После подключения зависимостей в сервисе будут доступны ендпоинты для работы с фронтендовой частью компоненты. Обязательно необходимо реализовать интерфейс авторизации для гибких таблиц и при необходимости ограничители данных (см. далее).

База данных сервиса должна содержать таблицы, в которых сохраняются настройки.

## Структура хранения для гибких таблиц

### *CMN_GRID_VIEW — настройка таблицы*

| **Колонка** | **Комментарий** |
|---|---|
| ID | Идентификатор |
| ENTITY_BRIEF | Сокращение сущности настройки |
| GRID | Журнал. Является дополнительным обязательным классификатором. Для одной и той же сущности может быть несколько разных по функциональности журналов, и для каждого журнала будет свой набор |

| Колонка | Комментарий |
|---|---|
| | настроек. |
| BIZ_PROCESS | Дополнительный необязательный классификатор, который позволяет для журнала сформировать отдельные настройки в рамках какого-то бизнес-процесса. |
| NAME | Наименование настройки. |
| KIND | Вид настройки, см.далее. |
| KIND_SPECIFIER | Спецификатор вида настройки, см.далее. |
| IS_DEFAULT | Является настройкой по умолчанию. |
| MANDATORY_FILTER | Требуется обязательный фильтр. |
| CREATE_DATETIME | Дата/время создания настройки. |
| CREATE_USER | Пользователь, создавший настройку. |
| UPDATE_DATETIME | Дата/время изменения настройки. |
| UPDATE_USER | Пользователь, изменивший настройку. |
| STRUCTURE | Структура настройки (JSON). |

### *CMN_GRID_VIEW_VALUE — значения фильтров таблицы*

| Колонка | Комментарий |
|---|---|
| ID | Идентификатор фильтра. |
| GRID_VIEW_ID | Идентификатор настройки. |
| USER_NAME | Имя пользователя. |
| VALUES | Значения фильтра (JSON). |

## Аннотация сущностей

Для того, чтобы работал универсальный механизм выборки данных, необходимо дополнительно аннотировать сущности, которые будут участвовать в выборке.

Сама сущность должна быть аннотирована с помощью `@EntityInfo` Аннотация имеет следующие поля:

- `String name();` Пользовательское наименование сущности, доступное в интерфейсе. Обязательное поле. Пример: `name = "Договор страхования"`
- `String namingAttributes() default "";` Атрибуты, входящие в «Наименование объекта». Перечисляются через запятую. «Наименование объекта» — вычисляемый псевдоатрибут, который состоит из нескольких атрибутов, разделённых пробелами, и позволяет охарактеризовать объект для пользователя. Для договора в «Наименование объекта» могут входить серия и номер договора. Для физического лица в «Наименование объекта» могут входить фамилия, имя и отчество. Необязательное поле, пустое по умолчанию. Пример: `namingAttributes = "contrSeries,contrNumber"`
- `boolean isTreeLike() default false;` Логическое. **ВНИМАНИЕ! Данная опция пока недоступна!** Сущность является древовидной (иерархичной). Для древовидных сущностей будут доступны способы получения потомка или наследников. Необязательное поле, false по умолчанию. Пример: `isTreeLike=true`

Например:

```
@Entity
@Table(name = "B2B_CONTR")
@EntityInfo(name = "Договор страхования", namingAttributes =
"contrPolSer,contrPolNum")
@CustomDataRestriction
```

```
public class Contract extends PanacheEntityBase {
```

В данном примере объявляется сущность “Договор страхования”, у которой наименование объекта составляется из двух атрибутов (серия полиса и номер полиса), а также сущность имеет пользовательское ограничение данных.

В рамках сущности поля должны быть аннотированы с помощью `@AttributeInfo` Аннотация имеет следующие поля:

- `String name();` Пользовательское наименование атрибута. Обязательное поле. Пример: `name = "Номер договора"`
- `AttributeType type() default AttributeType.FIELD;` Тип атрибута: `AttributeType.FIELD` — поле, `AttributeType.RELATION` — ссылка. Необязательное поле, `AttributeType.FIELD` по умолчанию. Пример: `type = AttributeType.RELATION`

Дополнительное пояснение. Для атрибутов типа ссылка возможны два варианта:

1. Ссылка через тип. Т.е. тип поля класса является другим классом. Дополнительно указывать ссылочный класс через refEntity не требуется. Например:
```
@ManyToOne(fetch = FetchType.EAGER)
@JoinColumn(name = "PRODID")
@AttributeInfo(name = "Страховой продукт", type = AttributeType.RELATION)
public Product product;
```
2. Ссылка через идентификатор (типа Long). Требуется дополнительно указывать ссылочный класс через refEntity. Например:
```
@Column(name = "PRODID")
@AttributeInfo(name = "Страховой продукт", type = AttributeType.RELATION,
refEntity = "Product")
public Long productId;
```

Продолжение списка полей `@AttributeInfo`:

- `AttributeKind kind() default AttributeKind.DB;` Вид атрибута: `AttributeKind.DB` - поле из БД; `AttributeKind.CALC_DB` - поле, вычисляемое в БД; `Kind.CALC_JAVA` - поле, вычисляемое в БД/микросервисе. Необязательное поле, AttributeKind.DB по умолчанию. Пример: `type = AttributeKind.DB`
- `String refEntity() default "";` Сокращение сущности, на которую ссылается атрибут. Применимо только для типа AttributeType.RELATION. Необязательное поле, пустое по умолчанию. Пример: `refEntity = "ProductVersion"`
- `boolean isJournalable() default true;` Признак, что атрибут может выводиться в журнал. Не рекомендуется допускать вывод в журнал “тяжелых” атрибутов типа memo-поле или изображение. Необязательное поле, true по умолчанию. Пример: `isJournalable = false`
- `boolean isFilterable() default true;` Признак, что в журнале можно фильтровать по значению атрибута. Рекомендуется допускать фильтрацию только по ограниченному набору атрибутов в целях снижения нагрузки на базу данных. Необязательное поле, true по умолчанию. Пример: `isFilterable = false`
- `boolean isSortable() default true;` Признак, что в журнале можно сортировать по значению атрибута. Нельзя сортировать по memo-полям или изображению. Необязательное поле, true по умолчанию. Пример: `isSortable = false`
- `int defaultWidth() default 100;` Ширина по-умолчанию для отображения колонки с атрибутом в журнале в пикселях. 100 px по умолчанию. Необязательное поле, пустое

по умолчанию. Пример: `defaultWidth = 150`

Например:

```
    @Id
    @Column(name = "CONTRID")
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator =
"B2B_CONTR_SEQ")
    @SequenceGenerator(name = "B2B_CONTR_SEQ", sequenceName = "B2B_CONTR_SEQ")
    @AttributeInfo(name = "Ид")
    public Long contrId;

    @Column(name = "CONTRNUMBER")
    @AttributeInfo(name = "Полный номер договора")
    public String contrNumber;

    @Column(name = "CANCELDATE")
    @Convert(converter = DoubleDateConverter.class)
    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern =
JSONConstants.DateFormat)
    @AttributeInfo(name = "Дата расторжения договора")
    public Date cancelDate;

    // Вычисляемы атрибут
    @Transient
    @AttributeInfo(name = "Есть организации",
            kind = AttributeKind.CALC_DB,
            expressionSelect = "case when exists (select 1 from
ContractOrgStructure cos where cos.contrId = [contrId]) then 1 else 0 end",
            conditionEqual = "exists (select 1 from ContractOrgStructure cos
where cos.contrId = [contrId])",
            conditionNotEqual = "not exists (select 1 from ContractOrgStructure
cos where cos.contrId = [contrId])",
            isSortable = false)
    public boolean hasOrgs;

    @Column(name = "PRODVERID")
    @AttributeInfo(name = "Версия продукта", type = AttributeType.RELATION,
refEntity = "ProductVersion")
    public Long prodVerId;
```

### *Расширенные атрибуты*

Это особенность проекта B2B в СБС. Описания атрибутов хранятся в таблицах INS_HBDESCR, INS_HBPROPDESCR, INS_HBDATAVER. Значения расширенных атрибутов (как правило) хранятся в таблицах B2B_CONTREXT, INS_HBSTORE. Актуально только для проектов СБС на базе структуры таблиц B2B.

Нотация расширенных атрибутов:

“{ИМЯ_РСАШИРЕННОГО_НАБОРА:ИМЯ_АТРИБУТА_В_НАБОРЕ}”

где:

- ИМЯ_РСАШИРЕННОГО_НАБОРА - поле NAME из таблицы INS_HBDESCR.
- ИМЯ_АТРИБУТА_В_НАБОРЕ - поле NAME из таблицы INS_HBPROPDESCR.

Например:

- “{B2B.FlatProtection.ExternalPartners.Contr.Values:flatVOiIOInsAmount}” Расширенные атрибуты для продукта “B2B. Защита квартиры внешние партнёры. Показатели по договору”, атрибут “Страховая сумма ВОиИО”.
- “{B2B.HomeProtection.ExternalPartners.Contr.Values:privateHouseVOiIOInsAmount}”

Расширенные атрибуты для продукта “B2B. Защита дома внешние партнёры. Показатели по договору”, атрибут “Страховая сумма ВОиИО”.

Каждый атрибут каждого расширенного набора отображается в своей колонке в гибкой таблице. Однако, как в примере выше, логично для похожих атрибутов объединять колонки с атрибутом, который означает одно и то же значение. Для этого используется группировка расширенных атрибутов со следующей нотацией:

“{*:[Страховая сумма ВОиИО]}”

т.е. все расширенные наборы, в которых есть атрибут “Страховая сумма ВОиИО”.

## Ограничения видимости данных

Следующие методы получения данных являются универсальными:

- `/data2/list` метод получения объектов сущности; в параметрах указывается перечень требуемых атрибутов, критерии выбора данных, сортировка;
- `/data/list` метод получения объектов сущности согласно настройке гибкой таблицы.

Их универсальность означает, что путём прямого вызова таких методов можно получить большое количество данных. Логично, что накладывать ограничения для видимости данных при помощи критериев в данных методах нельзя.

Чтобы пользователь получал только те данные, к которым он имеет право доступа, реализован механизм ограничителей — DataRestrictor. На данный механизм нельзя повлиять извне.

Предполагается создать разные ограничители видимости данных, например: - по организационной структуре; - только своих данных; - по роли и т. п.

### *Стандартный ограничитель видимости данных*

Стандартный ограничитель видимости данных позволяет ограничивать данные в разрезе организационно-штатной структуры.

Сначала необходимо сущность аннотировать как `@StandardDataRestriction`, что означает, что есть стандартное ограничение данных для сущности.

У аннотации есть 2 поля, в которых указываются поля в таблице базы данных:

- `String orgStructureIdFieldName();` Наименования поля с идентификатором орг.структуры.
- `String productIdFieldName();` Наименования поля с идентификатором продукта.

Если поле не указано или пустое, то в данном разрезе ограничение не накладывается.

Например:

```
@Entity
@Table(name = "B2B_CONTR")
@EntityInfo(name = "Договор страхования", namingAttributes = "contrId")
@StandardDataRestriction(orgStructureIdFieldName="orgStructureId",
productIdFieldName = "")
public class Contract extends PanacheEntityBase {
```

### *Кастомный ограничитель видимости данных*

Кастомный ограничитель позволяет реализовать произвольное ограничение, в котором условие ограничения создаётся в реализации интерфейса `com.bivgroup.lib.entitydata.data.CustomDataRestrictor`.

Сначала необходимо сущность аннотировать как `@CustomDataRestriction`, что означает, что есть кастомное ограничение данных для сущности, например:

```
@Entity
@Table(name = "B2B_CONTR")
@EntityInfo(name = "Договор страхования", namingAttributes = "contrId")
@CustomDataRestriction
public class Contract extends PanacheEntityBase {
```

`CustomDataRestriction` переводится как «Кастомное ограничение данных»

Далее требуется реализовать бин с интерфейсом `CustomDataRestrictor`.

`CustomDataRestrictor` переводится как «Кастомный ограничитель данных».

Данный бин должен иметь аннотацию `CustomDataRestrictorFor`, указывающую, на какую сущность накладывается ограничение, т. к. для разных сущностей ограничения могут быть разными.

`CustomDataRestrictorFor` переводится как «Кастомный ограничитель данных для сущности …».

В реализации интерфейсе необходимо переопределить 2 метода:

- `public String getJoinHQL();` Метод добавляет join в секцию FROM основного запроса.
- `public String getWhereHQL();` Метод добавляет условие через оператор AND в секцию WHERE основного запроса.

Если метод возвращает null, то основной запрос не модифицируется.

При помощи конструкции MainAlias можно обращаться к основной сущности. При подстановке запроса MainAlias будет заменён на необходимый правильный алиас.

Например:

```
@ApplicationScoped
@Unremovable
@CustomDataRestrictorFor(entity = "Contract")
public class ContractDataRestrictor implements CustomDataRestrictor {

    @Inject
    SBSAuthContextImpl auth;

    @Inject
    EntityQueryBuilderContext eqbContext;

    @Override
    public String getJoinHQL() {
        List<Long> orgStructureIds = auth.getOrgStructureIds();
        if (orgStructureIds.isEmpty()) {
            return null;
        } else {
            return "inner join ContractOrgStructure COS on (MainAlias.contrId =
COS.contrId)"
                    + " inner join InsDepLvl DEPLVL on (COS.orgStructId =
```

```
DEPLVL.objectId)";
        }
    }

    @Override
    public String getWhereHQL() {
        //if (!eqbContext.hasWhereConditions()) {
        //    ...
        //}

        List<Long> orgStructureIds = auth.getOrgStructureIds();
        switch (orgStructureIds.size()) {
            case 0:
                return null;
            case 1:
                return "DEPLVL.parentId = " +
Long.toString(orgStructureIds.get(0));
            default:
                String inStr =
orgStructureIds.stream().map(String::valueOf).collect(Collectors.joining(","));
                return "DEPLVL.parentId in (" + inStr + ")";
        }
    }

}
```

Особенность: обязательная аннотация @Unremovable, т. к. при её отсутствии бин будет удалён при сборке сервиса. Библиотека entity-data динамически осуществляет поиск бина, соответствующего сущности, и его вызов.

### *Контекст кастомного ограничителя видимости данных*

В ходе реализации одной из задач возникла необходимость понять, существуют ли в запросе ограничения, наложенные гибкой таблицей или принудительно. Для это было введено понятние контекста кастомного ограничителя видимости данных.

EntityQueryBuilderContext - RequestScoped-бин, который имеет метод
`public boolean hasWhereConditions();`

В результате вызова данного метода можно определить, есть ли какие-то ограничения в запросе.

Ограничение: предполагаем, что в рамках одного http-вызова обрабатывается только одно построение запроса и создаётся один экземпляр EntityQueryBuilder.

## Авторизация (права)

Для работоспособности компоненты необходимо отдельно реализовать интерфейс администрирования компоненты `com.bivgroup.lib.entityjournal.auth.AuthContext`.

Без реализации данного интерфейса сервис не соберётся.

Лучше данный интерфейс реализовать в отдельной библиотеке, которая должна быть общей на все сервисы в рамках Заказчика с единой точкой администрирования настроек.

Методы интерфейса:

- `String getUser();` Получить текущего пользователя.
- `Set<String> getUserGroups();` Получить группы, в которые входит текущий пользователь.

- `boolean userIsGridViewAdmin();` Текущий пользователь является администратором настроек гибких таблиц.
- `boolean userCanUpdateOwnGridView();` Текущий пользователь НЕ является администратором настроек гибких таблиц и может редактировать свои настройки.

Для простого тестирования компоненты можно реализовать заглушку для данного интерфейса: pom.xml:

```
        <dependency>
            <groupId>com.bivgroup.lib</groupId>
            <artifactId>entity-journal</artifactId>
            <version>${revision}</version>
        </dependency>
```

Класс TestAuthContextImpl

```
package com.bivgroup.lib.entityjournal.auth;

import java.util.HashSet;
import java.util.Set;
import javax.enterprise.context.ApplicationScoped;

/**
 * Тестовая реализация интрфейса для получения информации о пользователе и его
правах на гибкие таблицы.
 */
@ApplicationScoped
public class TestAuthContextImpl implements AuthContext {

    @Override
    public String getUser() {
        return "THE_USER";
    }

    @Override
    public Set<String> getUserGroups() {
        Set<String> result = new HashSet<>();
        return result;
    }

    @Override
    public boolean userIsGridViewAdmin() {
        return true;
    }

    @Override
    public boolean userCanUpdateOwnGridView() {
        return true;
    }
}
```

## Параметры в application.properties

В application.properties необходимо указать следующие параметры:

1. com.bivgroup.entitydata.useExternalDataSource=false Использовать внешние источники данных. true/false Внешний источник данных - это другой микросервис, который поставляет объекты той же сущности, что и в текущем микросервисе. Выбираемые наборы данных по сущности в текущем и внешнем микросервисах складываются при выдаче пользователю в гриде.
2. com.bivgroup.entitydata.externaldatasource.ENTITY.SVC_NAME=URL Указание

конкретного внешнего источника данных, где: ENTITY - имя класса сущности. SVC_NAME - имя сервиса. URL - url к сервису. Пример: com.bivgroup.entitydata.externaldatasource.Contract.autoapp=http://localhost:8012

3. com.bivgroup.entityjournal.grids.ENTITY=GRID_1,GRID_2,…GRID_N Указание списка журналов для сущности, где: ENTITY - имя класса сущности. GRID_1,GRID_2,… GRID_N - перечисленные через запятую и без пробелов имена журналов. Соответствуют полю GRID в таблице CMN_GRID_VIEW. Пример: com.bivgroup.entityjournal.grids.Contract=main
4. com.bivgroup.entityjournal.max.rows=N Установка максимального числа записей для вывода в гибкую таблицу для всех сущностей и журналов. N - целое число (по умолчанию 500).
5. com.bivgroup.entityjournal.max.export.rows=N Установка максимального числа записей для экспорта из гибкой таблицы для всех сущностей и журналов. N - целое число (по умолчанию 25000).
6. com.bivgroup.entityjournal.max.rows.ENTITY=N Установка максимального числа записей для вывода в гибкую таблицу для конкретной сущности. Имеет более высокий приоритет по отношению к com.bivgroup.entityjournal.max.rows. ENTITY - имя класса сущности. N - целое число.
7. com.bivgroup.entityjournal.max.export.rows.ENTITY=N Установка максимального числа записей для экспорта из гибкой таблицы для конкретной сущности. Имеет более высокий приоритет по отношению к com.bivgroup.entityjournal.max.export.rows. ENTITY - имя класса сущности. N - целое число.
8. com.bivgroup.entityjournal.max.rows.ENTITY.GRID=N Установка максимального числа записей для вывода в гибкую таблицу для конкретной сущности и конкретного журнала. Имеет более высокий приоритет по отношению к com.bivgroup.entityjournal.max.rows.ENTITY и com.bivgroup.entityjournal.max.rows. ENTITY - имя класса сущности. GRID - журнал. N - целое число.
9. com.bivgroup.entityjournal.max.export.rows.Contract.main=N Установка максимального числа записей для экспорта из гибкой таблицы для конкретной сущности и конкретного журнала. Имеет более высокий приоритет по отношению к com.bivgroup.entityjournal.max.export.rows.ENTITY и com.bivgroup.entityjournal.max.export.rows. ENTITY - имя класса сущности. GRID - журнал. N - целое число.

## *Документация для Front-End разработчика*

## Описание

Гибкие таблицы - Компонента предназначена для отображения данных в табличных формах (журналах).

Данная Front-End библиотека `lib.entity-journal.web` работает совместно с Back-End библиотекой `lib.entity-journal.backend`

### *Сборка*

Сборка библиотеки происходит в ветках со свойством “protected” Публикация в Nexus БИВ происходит автоматически BIV registry=https://192.168.1.46:8443/repository/microfrontend

Публикация в Nexus СБС происходит только в ручном режиме SBI registry=https://cpnexus01.sberins.ru/nexus/repository/biv-npm

Для сборки используется angular-runner, который работает на node:18

### *Подключение к проекту*

Библиотека доставляется на клиентские хранилища пакетов (например: Sonatype Nexus Repository) в виде npm-пакета.

Версия пакета выбирается исходя из версии Angular.
Например: если используется `Angular v17`, то необходимо выбрать версию `@biv/entity-journal v17.x.x`

Библиотека использует компоненты Angular Material

### *Подключение в компонете*

`example.component.ts`:

```
import { Component } from '@angular/core';
import { BivEntityJournalModule } from '@biv/entity-journal';

@Component({
    selector: 'app-example01',
    templateUrl: './example01.component.html',
    styleUrl: './example01.component.scss',
    standalone: true,
    imports: [
        BivEntityJournalModule
    ]
})
export class ExampleComponent { }
```

### *@Input параметры*

| **Параметр** | **Тип** | **Описание** | **Пример** |
|---|---|---|---|
| `api` | string | end-point микросервиса журнала | `"/api/b2b-journal"` |
| `entity` | string | имя сущности (таблицы) | `"Contract"` |
| `name` | string | произвольное имя для группировки настроек | `"main"` |
| `label` | string | пользовательское название журнала, отображаемое на интерфейсе | `"main"` |
| `filter` | JournalFilter | предустановленный(программный) фильтр | `{ operator: "AND", predicates: [ { attr: "contrId", operator: "IN", value: [ 10061623341, 10061641523 ] } ] }` |
| `customFilterUI` | boolean | Пользовательская реализация фильтра. Библиотека не будет | `false` \| `true` |

| Параметр | Тип | Описание | Пример |
|---|---|---|---|
| | | отображать встроенный интерфейс фильтра. | |
| `waitingFilter` | boolean | Журнал не загружает данные, пока не установлен любой из фильтров, пользовательский или программный. | `false` \| `true` |
| `waitingProgramFilter` | boolean | Журнал не загружает данные, пока не установлен программный фильтр. | `false` \| `true` |
| `fields` | Record<string, string> | список полей, которые необходимо включить в выборку | `{ "unique-field-name-alias1": "original-field-name" }` |
| `selectable` | boolean \| “single” \| “multiple” \| “range” | выключение включение режима выбора записей (см. `@Output` параметры rowSelect и selectionChanged) | `false` \| `true` \| `"single"` \| `"multiple"` \| `"range"` |
| `hideToolbar` | boolean | Скрыть панель инструментов | `false` \| `true` |
| `hideConfigurationList` | boolean | Скрыть панель с выбором конфигурации | `false` \| `true` |
| `hideCopyToClipboardButton` | boolean | Скрыть кнопку копирования в буфер обмена данных журнала (**начиная с версии 17.5.4**) | `false` \| `true` |
| `hideExportButton` | boolean | Скрыть кнопку экспорта данных журнала | `false` \| `true` |
| `hidePager` | boolean | Скрыть панель с пейджером | `false` \| `true` |
| `showFilterbar` | boolean | Отображать вспомогательную панель с установленными фильтрами | `false` \| `true` |
| `cellQuickSearch` | boolean | Быстрый поиск по полям таблицы, при клавиатурном вводе на поле | `false` \| `true` |
| `keepState` | boolean | Сохранять состояния журнала | `false` \| `true` |
| `keepMode` | `QueryParams` \| `LocalStorage` \| `SessionStorage` | Режим хранения состояний: `QueryParams` - в виде | `keepMode="LocalStorage"` |

| Параметр | Тип | Описание | Пример |
|---|---|---|---|
| | | сжатой строки в query-параметре `_sjsk` адресной строки страницы; `LocalStorage` и `SessionStorage` - в соответствующих хранилищах браузера. Если установлен `keepState` и не задан `keepMode`, то принимается значение по умолчанию `SessionStorage` | |
| `copyToClipboard` | boolean | Включить возможность копирования данных таблицы в буфер обмена (**начиная с версии 17.5.4**) | `false` \| `true` |
| `copyHeaderToClipboard` | boolean | Копировать в буфер обмена заголовки (**начиная с версии 17.5.4**) | `false` \| `true` |

### *@Output параметры*

| Параметр | Тип | Описание |
|---|---|---|
| `configurationList` | Array | Список доступных конфигураций |
| `configuration` | JournalConfiguration | Выбранная конфигураця |
| `rowClick` | <T> | Событие нажатия на строку с объектом записи |
| `rowDblClick` | <T> | Событие двойного нажатия на строку с объектом записи |
| `cellRightClick` | CellContextMenuEvent<T> | Событие нажатия на ячейку |
| `rowSelect` | <T> | Событие выбора строки (одна строка) с объектом записи |
| `selectionChanged` | Array<T> | Событие изменения выбора (одна или много строк) с массвом объектов выбранных записей |
| `stateChange` | JOURNAL_STATE \| null | Изменение режима журнала (просмотр / настройка / фильтр) |
| `cellKeyDown` | CellKeyDownEvent | Событие клавиатурного нажатия на выделенной ячейке таблицы |
| `userFilter` | JournalFilter \| undefined | Установленный пользовательский фильтр |

### *Методы компонента `BivEntityJournalComponent`*

| Метод | Параметры | Описание |
|---|---|---|
| `reload` | - | Перезагрузить данные в журнале |
| `purge` | - | Очистить данные в журнале |
| `applyFilter` | (filter: JournalFilter \| undefined, merge: boolean = false) | Применить фильтр к журналу, если во второй параметр передан `true`, то библиотека объеденит переданный фильтр с ранее установленным |
| `loadConfiguration` | number? | Загрузить конфигурацию по ID. Если ID не указан, то загружается настройка по умолчанию |
| `deleteConfiguration` | number | Удалить настройку по ID |
| `copyPageToClipboard` | - | Скопировать видимые данные (страницу) в буфер обмена (**начиная с версии 17.5.5**) |
| `copyPageWithHeadersToClipboard` | - | Скопировать видимые данные (страницу) вместе с заголовками в буфер обмена (**начиная с версии 17.5.5**) |
| `copySelectedToClipboard` | - | Скопировать выбранные данные в буфер обмена (**начиная с версии 17.5.5**) |
| `copySelectedWithHeadersToClipboard` | - | Скопировать выбранные данные вместе с заголовками в буфер обмена (**начиная с версии 17.5.5**) |
| `export` | “CSV” \| “XLSX” | Скачать данные в виде файла |

### *Injection Tokens*

| Токен | Тип | Описание |
|---|---|---|
| BIV_SMART_JOURNAL_PAGE_SIZE | number | Кол-во записей на странице по умолчанию. Необходимо учитывать, что конфигурация, которая приходит от сервера имеет высший приоритет. |
| BIV_SMART_JOURNAL_PAGE_SIZE_LIST | Array<number> | Список для выбора кол-ва записей |

### *Injection Tokens библиотеки Tabulator*

> Поддержка данных токенов появилась в версии v17.5.3
> {.is-warning}

| Токен | Тип | Описание |
|---|---|---|
| TABULATOR_CLIPBOARD_MODE | boolean \| “copy” \| “paste” \| undefined | Документация |
| TABULATOR_CLIPBOARD_COPY_ROW_RANGE | RowRangeLookup \| undefined | Документация |

| Токен | Тип | Описание |
| --- | --- | --- |
| TABULATOR_CLIPBOARD_COPY_CONFIG | boolean \| AdditionalExportOptions \| undefined | Документация |

### *Элементы управления через выпадающее меню*

Если необходимо скрыть toolbar журнала, но не потерять доступ к элементам управления журналом, то можно использовать выпадающее меню. В шаблоне создаём кнопку-инициатор

```
<button mat-icon-button (click)="onMenu($event)">
    <mat-icon>settings</mat-icon>
</button>

<biv-entity-journal #myJournal
    [hideToolbar]="true"
    >
</biv-entity-journal>
```

В компоненте вызываем `state.showJournalMenu` и передаём туда событие мыши по click

```
...
@ViewChild('myJournal', { read: BivEntityJournalComponent })
public readonly journal1!: BivEntityJournalComponent<any>;
...
public onMenu(mouseEvent: MouseEvent): void {
    this.journal.state.showJournalMenu(mouseEvent);
}
...
```

## Возможные ошибки и их решения

```
Error: NG05105: Unexpected synthetic property @transitionMessages found. Please
make sure that:
  - Either `BrowserAnimationsModule` or `NoopAnimationsModule` are imported in
your application.
  - There is corresponding configuration for the animation named
`@transitionMessages` defined in the `animations` field of the
```

Решение: добавить `provideAnimationsAsync()` в `src/app/app.config.ts`

```
import { ApplicationConfig } from '@angular/core';
import { provideRouter } from '@angular/router';
import { provideAnimationsAsync } from
'@angular/platform-browser/animations/async';

import { routes } from './app.routes';


export const appConfig: ApplicationConfig = {
    providers: [
        provideRouter(routes),
        provideAnimationsAsync(),
    ]
};
```

## Стилизация

Для стилизации гибких таблиц под стили заказчика в библиотеке используются глобальные стили.
В компонентах, где указаны эти стили прописано `encapsulation: ViewEncapsulation.None`.
Дабы не возникло ситуации пересечения с другими стилями, ко всем глобальным стилям установлен префикс `biv-entity-journal`.

## Классы компонентов

### *Общие*

Данные классы не привязаны к конкретному компоненту библиотеки и используются на общих элементах

- `.biv-entity-journal__field`
  Общий класс для всех `<mat-form-field>`
- `.biv-entity-journal__button`
  Общий класс для всех `<button mat-button>`
- `.biv-entity-journal__button-flat`
  Общий класс для всех `<button mat-flat-button>`

### *Пейджер*

- `.biv-entity-journal-pager__field-block`
  Обвязка для блока, где `<mat-label>` находится вне блока `<mat-form-field>`
- `.biv-entity-journal-pager__field`
  Дополнительный класс для `<mat-form-field>`, где также указан `.biv-entity-journal__field`
- `.biv-entity-journal-pager__page`
  Класс для кнопки перехода на страницу, либо перелистывания (вперёд, назад)
- `.biv-entity-journal-pager__more`
  Класс для элемента многоточие в списке страниц

### *Панель настроек*

- `.biv-entity-journal-configuration-list`
  Контейнер со списом настроек и кнопкой добавить
- `.biv-entity-journal-configuration-list__item`
  Элемент списка настроек
- `.biv-entity-journal-configuration-list__item-selected`
  Выбранная настройка
- `.biv-entity-journal-configuration-list__item-menu`
  Контекстное меню настройки
- `.biv-entity-journal-configuration-list__item-menu-option`
  Пункт контекстного меню настройки
- `.biv-entity-journal-configuration-list__button-add`
  Кнопка добавления настройки

### *Экранная форма настройки*

- `.biv-entity-journal-configuration__button`
  Кнопки управления “Отмена”, “Сохранить”
- `.biv-entity-journal-configuration__button-close`
  Кнопка закрытия настройки конкретной колонки или фильтра

### *Экранная форма фильтра*

- `.biv-entity-journal-filter__button`
  Кнопки управления “Отмена”, “Применить”

### *Вспомогательная панель с фильтрами*

- `.biv-entity-journal__filterbar`
  Контейнер панели
- `.biv-entity-journal__filterbar-item`
  Контейнер элемента фильтра
- `.biv-entity-journal__filterbar-label`
  Лэйбл элемента (наименование поля фильтра)
- `.biv-entity-journal__filterbar-operator`
  Оператор элемента
- `.biv-entity-journal__filterbar-value`
  Значения элемента

## Клонирование проекта

Окружение для разработки находится тут

В проекте используются `git submodule`

Для клонирования проекта необходимо выполнить команду

```
$ git clone --recurse-submodules -b my-favorite-branch
ssh://git@gitlab.bivgroup.ru:2222/biv-platform/entity-journal/web-devkit.entity-
journal.git
```

> Необходимо заменить `my-favorite-branch` на нужную Вам ветку {.is-warning}

Данная команда склонирует проект вместе со всеми зависимостями.

## Настройка и запуск

1. Устанавливаем npm-пакеты

```
$ npm i
```

2. Если необходимо, настраиваем проксирование запросов в файле `proxy.conf.json` на нужный нам сервер

   > RTFM: Proxying to a backend server {.is-info}

3. Запускаем библиотеку в режиме разработки

```
$ npm run start:lib
```

4. Запускаем сам проект в режиме разработки

```
$ npm start
```

5. Открываем http://localhost:4200

## *Администрирование настроек ГТ*

Администрирование настроек ГТ позволяет Администратору видеть и изменять **все** настройки, а не только те, к которым предоставлен доступ конкретному пользователю. Это особенно актуально для настроек тех сущностей, к которым ограничен доступ и у пользователей нет прав на редактирование этих настроек, например, в сущности есть конфиденциальные данные. Администрирование позволяет редактировать, удалить и

клонировать конкретную настройку ГТ. При удалении выводится информация об удаляемой настройке для контроля и подтверждения или отмены удаления При клонировании администратор может изменить вид настройки, спецификатор вида (например другого пользователя), тип журнала и задать новое наименование настройки Администрирование настроек ГТ реализовано на `Server-Driven UI (SDI)` как библиотека и подключается к сервису, в котором необходимо администрировать определённые в нём сущности.

## Подключение администрирования настроек ГТ на backend

Подключение библиотеки:

```
<dependency>
    <groupId>com.bivgroup.lib</groupId>
    <artifactId>entity-journal-admin</artifactId>
    <version>${entity-journal-admin.version}</version>
</dependency>
```

где ${entity-journal-admin.version} - актуальная версия, на текущий момент 3.13-SNAPSHOT